VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2014

Jiří Littner

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Littner**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ABB s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Pavla Dráždilová, Ph.D.**

Konzultant bakalářské práce: Bc. Vladimír Brada

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 27. dubna 2014 .....

Rád bych na tomto místě poděkoval Bc. Vladimíru Bradovi, jakožto mému konzultantovi ve firmě, ve které jsem vykonával odbornou praxi, za pomoc pochopení nastalých složitostí, a také bych chtěl poděkovat Mgr. Pavle Dráždilové, Ph.D. za rady poskytnuté při vypracování této práce, které mi pomohly onu práci zhotovit v konečné podobě.

Abstrakt

Smyslem této práce je ukázat využití mých teoretických znalostí, nabytých ze školy, v praxi na konkrétních úkolech. Hlavním úkolem bylo vytvoření webové aplikace DataBroker Emulator, která slouží k testování nově vyvíjených, či přepracovaných aplikací, které budou z aplikace DataBroker získávat data, a ty budou dále aplikacemi zpracovávána. Druhým úkolem bylo vyvinout desktopovou aplikaci nazvanou DataBroker TestClient, kterou bude možné testovat DataBroker Emulator a samotný DataBroker. Posledním úkolem, na kterém jsem v rámci odborné praxe pracoval, byla aplikace ComplexityMatrixTool, která má za úkol vypočítat rizika projektů v těžařství.

Klíčová slova: JQuery, ASP.NET, ABB, C#, Databáze

Abstract

The purpose of this work is to demonstrate the use of my theoretical knowledge acquired in school in practice on specific tasks. The main task was to create a web application DataBroker Emulator, which is used to test newly developed or overworked applications, that obtain data from DataBroker application and this data will be processed by these applications. The second task was to develop a desktop application called DataBroker TestClient which can be tested DataBroker Emulator and DataBroker to. The final task on which I worked during the professional practice was ComplexityMatrixTool application, whose task is to calculate the risk in mining projects.

Keywords: JQuery, ASP.NET, ABB, C#, Database

Seznam použitých zkratek a symbolů

SQL	–	Structured query language
MVC	–	Model View Controller
WCF	–	Windows Communication Foundation
CSS	–	Cascading Style Sheets
HTML	–	Hyper Text Markup Language
TFS	–	Team Foundation Server
IIS	–	Internet Information Services
URL	–	Uniform Resource Locator
UML	–	Unified Modeling Language
ORM	–	Object-Relational Mapping

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Při výběru bakalářské práce jsem měl možnost vybrat si z mnoha zajímavých témat ke zpracování, ale dozvěděl jsem se také o možnosti vykonat odbornou praxi, z níž bych následně vypracoval výslednou zprávu. Tato možnost se mi jevila jako nejlepší volba, protože bych měl možnost dozvědět se, jak to v takové firmě funguje, dozvědět se o chodu společnosti, o tom, jak funguje vývoj softwaru v týmu a nabýt nových znalostí a zkušeností, které jsou nezbytné pro přijetí do nějaké firmy po dokončení studia. O této skutečnosti jsem se také v průběhu vykonávání odborné praxe přesvědčil. Při rozhodování, ve které firmě bych měl odbornou praxi absolvovat, jsem se rozhodoval mezi několika firmami a vybral jsem si společnost ABB s.r.o. Do doby, než jsem nastoupil do firmy, jsem neměl nejmenší ponětí, že se jedná o celosvětovou korporaci, která se zabývá mnoha obory.

V části vývojářského oddělení, do kterého jsem byl zařazen, se většina programátorů zabývala vývojem kontrolních systémů pro výrobky používané v ropném průmyslu. Má webová aplikace, kterou jsem zprvu vyvíjel je používána jako emulátor dat zasílaných řídícími systémy, které pracují s hodnotami pocházejícími například z onoho ropného průmyslu.

Současně s vývojem webové aplikace jsem vyvíjel desktopovou aplikaci testovacího klienta, díky které jsem měl možnost otestovat správnou funkčnost mé webové aplikace, ale také funkčnost aplikace nasazené v ostrém provozu, kterou jsem svou webovou aplikací simuloval.

Jako poslední projekt, na kterém jsem pracoval, v rámci odborné praxe, byl projekt na výpočet rizik projektů v těžařství, který byl vyvíjen pro švýcarského zákazníka. Tento projekt byl vytvořen jako webová aplikace, která byla vyvíjena pomocí technologie ASP.NET MVC 4 a je zde využita i JacvaScript knihovna jQuery.

V následující kapitole jsou uvedeny základní informace o společnosti společně s mým pracovním zařazením do ní. Následuje kapitola 3 obsahující harmonogram úkolů v časovém rozpětí, které jsem postupem vývoje řešil. Za touto kapitolou jsou popsány technologie použité při vykonávání mé odborné praxe. V kapitole 5 je řešena problematika samotného vývoje již zmíněných aplikací. Popis jednoho z projektů je rozepsán i v kapitole 6, za kterou jsou shrnuty mé schopnosti nabyté ve škole a uplatněné v praxi, za nimiž v poslední kapitole následuje celkové shrnutí odborné praxe.

2 Odborné zaměření firmy a popis pracovního zařazení studenta

2.1 Odborné zaměření firmy

Firma ABB s.r.o. je mezinárodní korporací, která se zabývá mnoha výrobními odvětvími, ale jejím primárním zaměřením je poskytování technologií pro energetiku a automatizaci po celém světě. Pole působnosti společnosti je tak velmi rozsáhlé. Společnost působí ve více než 100 zemích světa a zaměstnává více než 150 tisíc lidí. Jen v České republice má společnost zastoupení v 8 městech, v jejichž pobočkách zaměstnává téměř 3300 lidí. Několik poboček má i v Ostravě, z nichž jedné je součástí i softwarové oddělení, ve kterém pracuje asi 25 zaměstnanců. Softwarové oddělení v Ostravské pobočce se zabývá vývojem kontrolních systémů a nově se rozšiřuje i na vývoj informačních systémů. Aplikace jsou zde vyvíjeny dle agilní metodiky SCRUM.

2.2 Popis pracovního zařazení studenta

Při vstupním pohovoru jsem měl možnost se rozhodnout, pomocí jaké technologie bych chtěl vyvíjet software v průběhu odborné praxe. Jelikož jsem již při studiu inklinoval spíše k technologii .NET, konkrétně k vývoji informačních systémů pomocí technologie ASP.NET, snažil jsem se směrovat můj budoucí vývoj k této technologii. Při pohovoru jsem se rovněž seznámil s mým konzultantem Vladimírem Bradou, který mi lehce nastínil, co budu vyvíjet, a domluvili jsme se na příchodu do práce.

Po prvním příchodu do práce, jsem byl proškolen o bezpečnosti práce a seznámen se zásadami chování ve firmě. Poté jsem již počal s Vladimírem Bradou hlouběji konzultovat samotnou aplikaci, na které jsem vzápětí začal pracovat.

3 Harmonogram

V této kapitole jsou popsány úkoly, které mi byly postupně přiděleny při vývoji aplikací, jejichž zhotovení bylo náplní mé odborné praxe. Úkoly jsou rozepsány bodově v časových intervalech, ve kterých mi byly zadány.

25.7. - 28.7.

- seznámení s pracovištěm, bezpečnostní proškolení
- studium funkčnosti simulované aplikace
- připojení projektu k TFS, studium metodiky vývoje SCRUM
- vytvoření databáze, webové stránky, entit, napojení dat a jejich zobrazení na stránce
- vytvoření webové stránky pro detailní zobrazení jednotlivých dat (OPCItems)

12.8. - 16.8.

- zobrazení historických dat na detailní stránce
- vytvoření funkcionality pro úpravu dat na detailní stránce
- funkcionalita pro vytvoření nových dat
- implementace pro získávání dat přes WCF
- zobrazování popisu kvality signálu namísto číselného vyjádření
- vytvoření aplikace testovacího klienta
- zobrazení aktuálních dat v testovacím klientovi

19.8. - 23.8.

- možnost vymazání historických dat z detailní stránky
- implementace funkcionality pro vrácení hrubých historických hodnot pro jeden a více OPCItems z WCF služby aplikace DataBroker Emulator
- zobrazení hrubých historických dat pro jeden a více OPCItems v aplikaci testovacího klienta
- implementace podpory agregačních funkcí pro WCF službu DataBroker Emulátoru
- umožnění výběru a zobrazení agregovaných hodnot v aplikaci testovacího klienta
- úprava designu detailní stránky
- nastavení aplikace DataBroker Emulator pro běh na IIS namísto lokálního serveru programu Visual Studio

26.8. - 30.8.

- umožnění povolení a zakázání OPCItem na detailní stránce
- možnost filtrování dat a uchování filtrů
- oddělení ukládání konfigurace o OPCItems od jejich hodnot
- umožnění smazání a vytvoření nové hodnoty v historii
- přesunutí zdrojového kódu služby na straně klienta do oddělené třídy

2.9. - 6.9.

- umožnění zadat přihlašovací údaje pro připojení v testovacím klientovi
- konfigurace připojení v testovacím klientovi pomocí HTTP a NET.TCP vazby
- možnost připojování a odpojování testovacího klienta k aplikacím DataBroker a DataBroker Emulator

9.9. - 13.9.

- možnost seřazení dat v tabulce na hlavní stránce
- umožnění stránkování dat v tabulce na hlavní stránce
- funkcionalita pro zachování stránkování a seřazení při přechodu mezi stránkami
- stránkování historických dat na detailní stránce
- možnost změny historických dat

16.9. - 27.9.

- možnost rychlé změny hodnot jednotlivých OPCItems na hlavní stránce
- rozšíření historických dat o více režimů
- vytvoření jednoduchého generátoru dat

30.9. - 11.10.

- vytvoření souhrnné stránky
- zpoždění služby
- úprava řešení pro statickou OpcItem
- stromová hierarchie na detailní stránce

14.10. - 25.10.

- vytvoření vlastního ovládacího prvku tabulky
- úprava hlavní stránky
- vytvoření nové stránky se zobrazením OPCItems ve stromové hierarchii

28.10. - 1.11.

- podrobnější studium metodiky vývoje SCRUM
- analýza Excel sešitu
- návrh a vytvoření UML diagramu
- přidání do projektu nástrojů PetaPoco a RoundhousE

4.11. - 8.11.

- vytvoření databáze
- vytvoření modelů
- vytvoření CRUD operací
- vytvoření Controller
- vytvoření Views

11.11. - 15.11.

- funkcionalita v Controller pro administrátorskou část
- úprava validací v modelech

18.11. - 22.11.

- dynamické výpočty pro tabulku a pro graf
- uživatelský manuál

4 Technologie

V průběhu vykonávání odborné praxe jsem se setkal s některými technologiemi, které jsem již znal a se spoustou technologií nových, které jsem se potřeboval doučit. V této kapitole bych chtěl pár těchto technologií stručně popsat.

4.1 ASP.NET

ASP.NET je sada technologií v .NET frameworku pro vytváření webových aplikací a XML webových služeb vyvinutá ve firmě Microsoft. Stránky ASP.NET jsou spuštěny na serveru a generují kód značkovacího jazyka jako je například HTML, který je následně zaslán prohlížeči počítače nebo mobilního zařízení na straně klienta. Stránky ASP.NET používají zkompilovaný, událostmi řízený programovací model, který umožňuje oddělení aplikační logiky od uživatelského rozhraní. ASP.NET stránky a XML webové služby obsahují logiku na straně serveru, jež je psána v některém z programovacích jazyků kompatibilních s .NET frameworkem firmy Microsoft, jako je například jazyk Visual C# .NET. Více na webových stránkách firmy Microsoft, viz reference [2]

4.2 ASP.NET MVC framework

ASP.NET MVC framework vychází z návrhového vzoru Model-View-Controller (MVC), který je složen ze tří komponent, které mezi sebou komunikují. Jsou to model, view a controller. Tento framework je alternativou ASP.NET webových formulářů, které slouží, jak jsem již zmínil, pro tvorbu webových aplikací. ASP.NET MVC framework je lehký s jednoduchým testováním prezentační vrstvy, který má stejně jako aplikace založené na webovém formuláři integrovány existující ASP.NET funkce jako je například membership authentication. Programátor má také možnost kombinovat ASP.NET webový formulář společně s frameworkem ASP.NET MVC. Více na webových stránkách firmy Microsoft, viz reference [2]

4.3 Windows Communication Foundation

Windows Communication Foundation (WCF) je jednotný programovací model pro vytváření aplikací orientovaných na služby vyvinutý společností Microsoft. Tento model umožňuje vývojářům vytvářet bezpečné a spolehlivé transakční řešení napříč platformami. Používání WCF může programátor posílat data formou asynchronních zpráv z jednoho koncového bodu služby do druhého. Koncový bod služby může být součástí trvale dostupné služby hostované službou IIS, nebo to může být hostitelská služba v aplikaci. Zprávy zasálané službou WCF mohou být jednoduché, jako například jeden znak nebo slovo odeslané ve formátu XML nebo složité, jako třeba proud binárních dat. Více na webových stránkách firmy Microsoft, viz reference [2]

4.4 RoundhousE

Nástroj RoundhousE slouží k řízení změn a verzí databáze. Tento nástroj se snaží řešit údržbu a snadné nasazení databáze. RoundhousE si může programátor jednoduše přidat do řešení a v přidaném nástroji jsou složky pro vložení databázových skriptů. Složky jsou zde odděleny pro vložení skriptů tvořících uložené procedury, pro uložené funkce, pro vytvoření pohledů a složka pro postupnou úpravu databáze. Skripty ve složce pro úpravu databáze jsou vždy spuštěny jen jednou, což si nástroj řeší tak, že si do vlastní tabulky ukládá, které skripty již byly spuštěny. Když chce programátor databázi upravit po spuštění nástroje RoundhousE, musí vždy přidat nový skript s úpravou. Když upraví skript, který byl již spuštěn, nástroj zhavaruje a programátor je nucen tento problém složitě řešit nebo má možnost smazat celou databázi i s daty. Více na webových stránkách, viz reference [4]

4.5 PetaPoco

PetaPoco je lehký objektově-relační mapovač pro .NET aplikace. Důraz tohoto nástroje je kladen na jednoduchost použití a výkon, nikoliv na bohatost funkcí jako je tomu u jiných objektově-relačních mapovačů například u Entity Framework. PetaPoco je pouze jediný C# soubor. Když si programátor přidá do řešení PetaPoco, pouze mu nadefinuje připojovací řetězec a nastaví připojení k databázi a může hned vygenerovat ORM podle databázových tabulek, s jejímiž objekty může hned pracovat. Více na webových stránkách, viz reference [5]

5 DataBroker a jeho emulátor

Tato kapitola pojednává o aplikacích, které byly hlavní náplní mé odborné praxe. Mým hlavní úkolem odborné praxe bylo vytvořit emulátor aplikace DataBroker, proto jsou v první podkapitole uvedeny základní informace o této aplikaci a důvody, proč měla firma potřebu emulátor vyvinout. V druhé podkapitole jsou uvedeny základní informace funkčnosti mnou vyvíjené aplikace DataBroker Emulator a samotný popis řešení úkolů mi zadaných při vývoji. Poslední podkapitola obsahuje popis řešených úkolů při vývoji aplikace testovacího klienta, jež slouží pro otestování mnou vyvíjené aplikace i aplikace DataBroker.

5.1 DataBroker

DataBroker je služba, která pracuje jako most mezi řídícími systémy a aplikacemi zpracovávající data z řídících systémů. Komunikuje s řídícími systémy pomocí OPC protokolu. K této aplikaci je možné připojit libovolný řídící systém, který pracuje jako OPC server. Pro komunikaci s aplikacemi DataBroker využívá službu WCF.

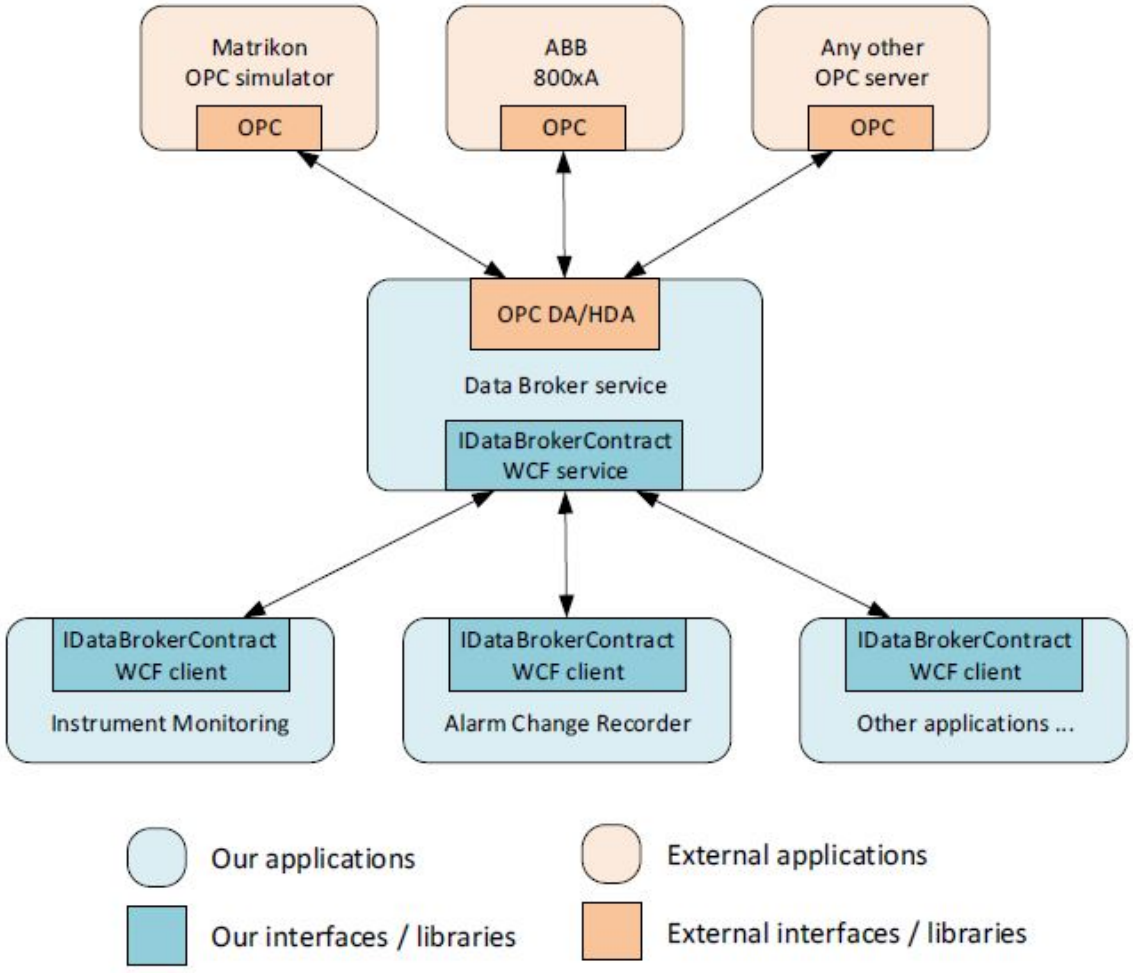
Doposud při vyvíjení aplikací, které budou připojeny na DataBroker, byl zapotřebí běžící DataBroker, aby šlo tyto aplikace otestovat. Před vytvořením DataBroker Emulatoru, vývojáři používali některé testovací servery, kde byl DataBroker nainstalován společně s řídícími systémy, ale:

- Je těžké simulovat data, které jsou potřebná pro vyvíjené aplikace
- Řídící systémy běží někde v síti, takže můžou být nedostupné
- Pouze jediné řešení pro lokální testování je virtualizace s celými řídícími systémy
- Řídící systémy nejsou pod kontrolou vývojářů

Kvůli výše uvedeným důvodům potřebovala firma vyvinout software DataBroker Emulator, který jim poskytne potřebná data, poběží lokálně a bude pod jejich kontrolou. Byl zapotřebí i testovací klient, který jim umožní zjistit, zda skutečný DataBroker pracuje správně.

5.1.1 Architektura aplikace

DataBroker je napsán jako webová aplikace pomocí framework .NET 4.0 (3.5 pro kontrakt), který běží pod IIS 7.0 a využívá WCF pro komunikaci s aplikacemi vyvíjenými pro práci s řídícími systémy. Používá ApcDaNet knihovnu od vedoucího poskytovatele OPC .NET komponent Advosol pro komunikaci s řídícími systémy. Umožňuje čtení a zapisování aktuálních a historických dat včetně agregací.



Obrázek 1: Vzájemná komunikace DataBrokeru s ostatními systémy

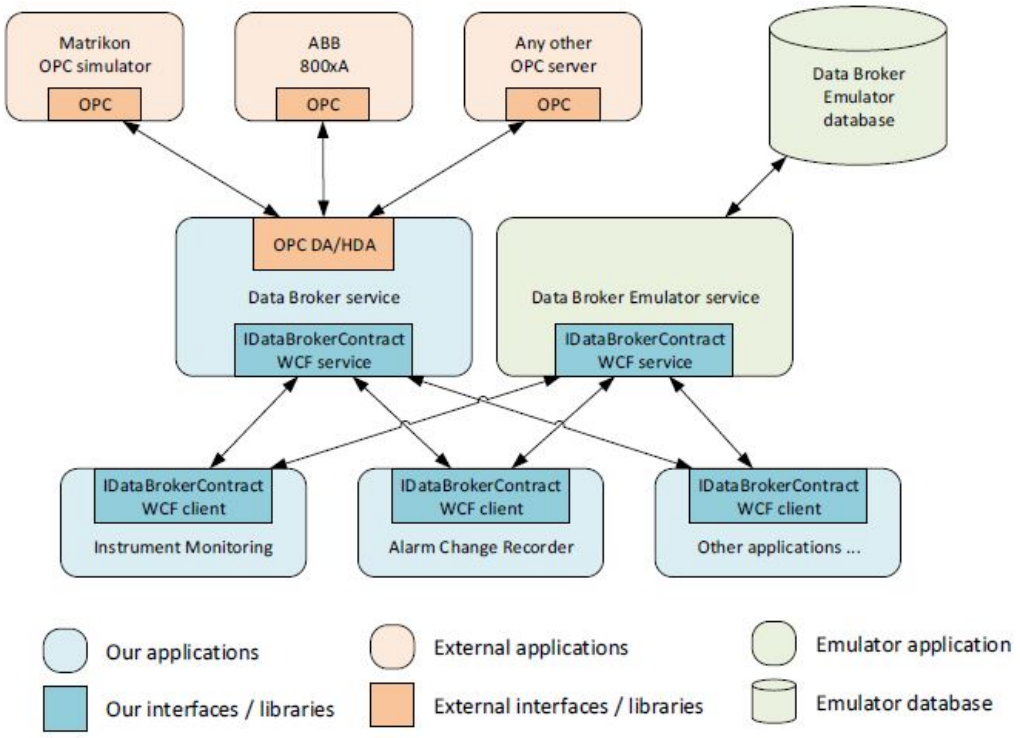
5.2 DataBroker Emulator

Aplikace DataBroker Emulator byla hlavní náplní mé práce při absolvování odborné praxe. Tato aplikace emuluje aplikaci DataBroker, jež sbírá data z OPC serverů a poskytuje je aplikacím, které tato data zpracovávají. Funkcionalita aplikací zpracovávající data z Opc serverů je rozšiřována, a také jsou vyvíjeny nové aplikace. Aby mohli vývojáři tyto aplikace otestovat před jejich nasazením, potřebovali vyvinout aplikaci DataBroker Emulator, která má vlastní databázi s daty (OpcItems a jejich hodnoty), která odpovídají datům zasílaným Opc servery. Samotná aplikace umožňuje zobrazení dat na webové stránce, jejich úpravu, vytvoření, zobrazení v jejich stromové struktuře a pochopitelně i zprostředkování těchto dat skrze WCF službu.

5.2.1 Architektura aplikace

Aby byla tato aplikace plně kompatibilní s aplikací DataBroker, protože vychází z její architektury, měl jsem při prvním příchodu připraveno pouze strukturované řešení, které již mělo přidáno kontrakty z aplikace DataBroker pomocí odkazů na knihovny .dll.

Aplikace je napsána jako webová aplikace pomocí framework .NET 4.0 (3.5 pro kontrakt), která běží pod IIS 7.0 a využívá WCF pro komunikaci s aplikacemi vyvíjenými pro práci s řídícími systémy, ale již nekomunikuje s řídícími systémy. Tato aplikace využívá své vlastní databáze se simulovanými daty.



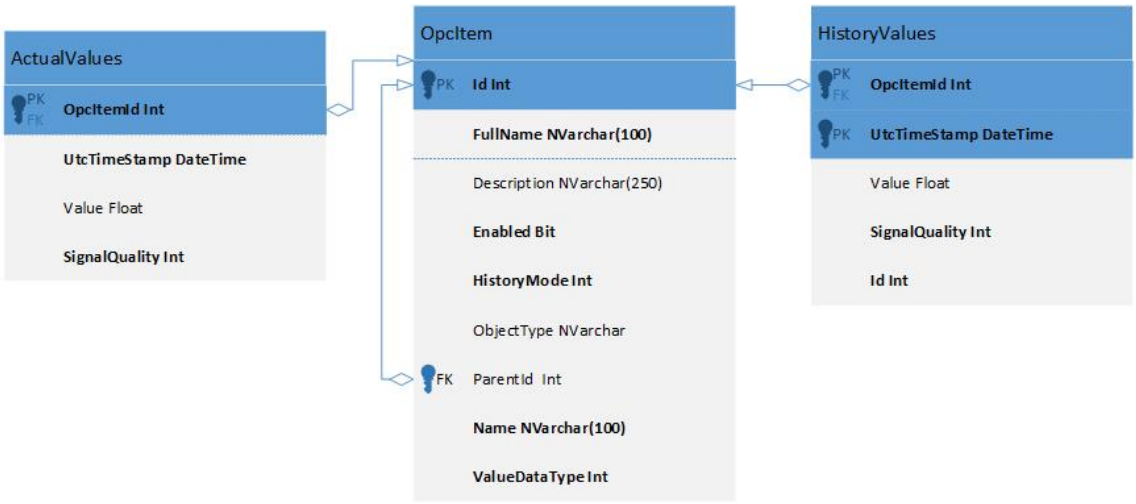
Obrázek 2: Architektura aplikace DataBroker Emulator

5.2.2 Řešení úkolů DataBrokerEmulátoru

Práci jsem počal seznámením se s pracovištěm a bezpečnostním školením. Následně jsem se seznámil s problematikou aplikace DataBroker Emulator, již jsem měl za úkol vytvořit. Seznámen jsem byl také s agilní metodikou vývoje SCRUM, viz reference [3], jejíž zjednodušenou formou jsme aplikaci vyvíjeli. Vývoj byl rozdělen do dvoutýdenních intervalů. Vždy po skončení jednoho intervalu jsem se dozvěděl, na jaké funkcionalitě budu pracovat v intervalu dalším. Velkou výhodou pro mne bylo, že v roli zákazníka byl můj konzultant, tudíž jsem s ním mohl jakoukoliv nejasnost konzultovat kdykoliv v průběhu vývoje. Po seznámení jsem si převzal předpřipravené řešení a konzultant si vytvořil vlastní TFS, ke kterému mi pomohl připojit mé řešení a vysvětlil mi, jak TFS funguje. V tuto chvíli jsem již byl ve fázi, kdy jsem počal samotnou implementaci, jejíž problematika je řešena v následujících podkapitolách. Aplikace se skládá ze čtyř webových stránek. Na hlavní stránce jsou zobrazeny všechny OpcItems s jejími aktuálními hodnotami a uživatel zde může data vyfiltrovat, seřadit, upravit a mazat. Z hlavní stránky může uživatel přejít na detailní stránku, která mu zobrazí informace o jedné OpcItem její historické hodnoty a také uvidí potomky dané OpcItem. Při připojení k aplikaci uživatel uvidí jako první souhrnnou stránku, ze které může přejít na hlavní stránku nebo na stránku stromové hierarchie, která slouží pouze k zobrazení OpcItems ve stromové hierarchii.

Databáze a zobrazení dat na hlavní stránce

Mou první prací bylo napsání skriptů pro vytvoření databáze, která se postupem vývoje rozšířila do podoby, již je zobrazena na obrázku 3. Databázové skripty jsou vloženy v nástroji RoundhousE, skrze který jsou skripty vždy spuštěny. Zde pro mne bylo novinkou vytvářet konkrétně primární klíč nikoliv přímo v definici tabulky, ale jako úprava tabulky a přidání pojmenované vazby primárního klíče.



Obrázek 3: Schéma databáze

Hlavní stránka

Jako první jsem vytvořil v projektu novou prázdnou stránku, kterou dále označuji jako hlavní. Abych mohl zobrazit data na této stránce, vytvořil jsem entity, které reprezentují objekty databázových tabulek, a vytvořil jsem jednoduchý náhled webové stránky, jenž zobrazuje OpcItems v ovládacím prvku GridView. Pro zobrazení dat jsem nejprve musel nadefinovat připojovací řetězec ke své lokální databázi a naimplementovat metody pro získání dat z databáze. Pro usnadnění přidávání parametrů k SQL dotazu (aby bylo možno zabránit napadení aplikace), mi konzultant poskytl třídu obsahující metody pro přidání parametrů všech typů. Po vybrání dat z databáze jsem potřeboval převést data z objektu SQLDataReader do objektů entit, a proto jsem vytvořil univerzální metody, které jsem nadále použil ve všech dalších metodách sloužících pro výběr dat z databáze. Data jsou na stránce zobrazena až po kliknutí na tlačítko z důvodu později přidané možnosti filtrování.

Uživateli jsem na stránce umožnil jednotlivé OpcItems mazat. Jelikož jedna OpcItem má data uložena ve všech databázových tabulkách, bylo nutné smazat záznamy ze všech tří databázových tabulek. Všechny SQL příkazy na mazání byly uzavřeny v jedné transakci, at' se provedou všechny najednou, jinak ani jeden.

Jelikož bude aplikace při používání obsahovat velké množství dat, nemohla být všechna data zobrazena na stránce najednou. Proto jsem v ovládacím prvku GridView umožnil stránkování a seřazení dat a na stránce jsem uživateli umožnil data filtrovat, což jsem vyřešil následující způsobem. Do metody pro výběr všech OpcItem z databáze jsem přidal parametry odpovídající argumentům filtrování a dále jsem přemýšlel, jak tyto argumenty přidat k SQL dotazu jako omezovací podmínku, tak aby byl SQL dotaz vždy sestaven správně, i když budou parametry do dotazu přidávány v jiném pořadí nebo když jsou všechny nebo jen některé úplně vynechány. Problém mi pomohl vyřešit konzultant, od kterého jsem dostal radu, at' k původnímu dotazu přidám klauzuli **WHERE** s podmínkou **1=1**, což v logice odpovídá tautologii. Tato podmínka neomezí výběr záznamů v původním dotazu a při přidávání podmínek omezujících výběr jsem pouze přidal podmínku k původnímu dotazu pomocí **AND**. Teď již nezáleží na pořadí, ani jestli k dotazu přidám nějakou podmínku. Podmínky jsou k SQL dotazu přidávány pouze tehdy, jestli parametry, které metoda přijímá, mají hodnotu nebo nejsou prázdné. Možnost použití tzv. WildCard znaků ve filtrování, jsem vyřešil metodou Replace, která nahradila běžně používané WildCard znaky v proměnných za WildCard znaky, které se používají při dotazování na databázi.

Pro zjednodušení práce při vytváření funkcionality pro seřazování a stránkování dat v GridView jsem přidal do řešení ovládací prvky DataSource a SortExtender, které mi poskytl konzultant, a které si ve firmě naprogramovali. Přidané prvky jsem pouze připojil k prvku GridView a nastavil jsem podle jakého atributu a kolik OpcItems na stránce má být v GridView zobrazeno. Tímto jsem umožnil stránkování a seřazení dat, již je zobrazeno na obrázku 4.

Aby byly filtry, prohlížená stránka a seřazení uchovány i po přechodu na detailní stránku a zpět, vytvořil jsem konstanty obsahující klíčové slovo označující tyto argumenty. Tyto konstanty označují názvy jednotlivý Session, do kterých jsou argumenty uloženy. Při vrácení se zpět na hlavní stránku jsou argumenty znovu získány ze Session a podle nich je tabulka načtena.

Na této stránce má uživatel také možnost měnit aktuální hodnoty přímo v ovládacím prvku GridView. Po kliknutí na tlačítko Edit jsou všechny záznamy procházeny v cyklu a ve sloupcích zobrazující aktuální hodnoty jsou změněny prvky popisku na textová pole a rozbalovací seznamy, které již mají načtena data a vybrány aktuální hodnoty. Při kliknutí na tlačítko Cancel jsou načteny filtry ze Session, vyfiltrována data a všechny prvky popisků jsou zobrazeny zpět i se svými původními daty. Při kliknutí na tlačítko Save jsou procházeny všechny řádky tabulky a hodnoty jsou uloženy do databáze. Po uložení jsou opět načteny filtry a vyfiltrovaná data jsou zobrazena v GridView. Jelikož jsem přišel na to, že při zobrazení jiné stránky tabulky nezůstane tabulka v editovacím módu, vytvořil jsem privátní vlastnost IsEditMode, jež je zobrazena ve výpisu 1 zdrojového kódu.

```
private bool IsEditMode
{
    get
    {
        if (ViewState["EditMode"] is bool)
            return (bool)ViewState["EditMode"];
        return false;
    }
    set { ViewState["EditMode"] = value; }
}
```

Výpis 1: Vlastnost IsEditMode

Tato vlastnost je naplněna při kliknutí na některé z tlačítek a její uložená hodnota je předána vlastnosti Cancel argumentu události přechodu stránky v GridView. U OpcItems, které mají nastaven mód historie *Static* se neupravuje aktuální hodnota, ale aktuální hodnotou je poslední historická hodnota. Stránka v editovacím módu je zobrazena na obrázku 4.

OPC ITEMS

Name: *d_5* Description: *D* History mode: Enabled: Yes Object type:

Select into Grid

New Item | Cancel | Save

Id	Full name	History mode	Description ▲	Enabled	Time stamp	Value	Signal quality	Object type	Edit item	Delete item
395	da517834ce7549f	Static	AELNVZJTDNRFQJVTLZXMSQSSLAXVXV	Yes	23.10.2013 12:59:24	4418	80 - Sensor Calibration		Details	Delete
338	324d556b7e0d4b6	Dynamic periodical	CFBAKMSJEDBAUZGPVVACTDWOBUMTSH	Yes	27.10.2013 12:59:24	21803	16 - Sensor Failure		Details	Delete
194	2bdc580043a1434	Static	IFOLRGAWULEBDLEGOEYMYRQKYKQIXS	Yes	30.10.2013 12:59:23	385	12 - Device Failure		Details	Delete
430	dbf21961f41d456	Disabled	MZMVGHHZLBPAWOAEASDXLXBFSJAMZM	Yes	30.10.2013 7:43:32	28691	16 - Sensor Failure		Details	Delete
465	4e55de525d094a4	Dynamic on change	UQWNROHSXDRPVONGOFNHBOEWZTMYND	Yes	19.10.2013 12:59:24	26576	24 - Comm Failure		Details	Delete
471	9edd6ddf55744a1	Disabled	ZTNDRQEGTONUDSGPPKJHEELSARDDCC	Yes	1.10.2013 12:59:24	17292	20 - Last Known		Details	Delete

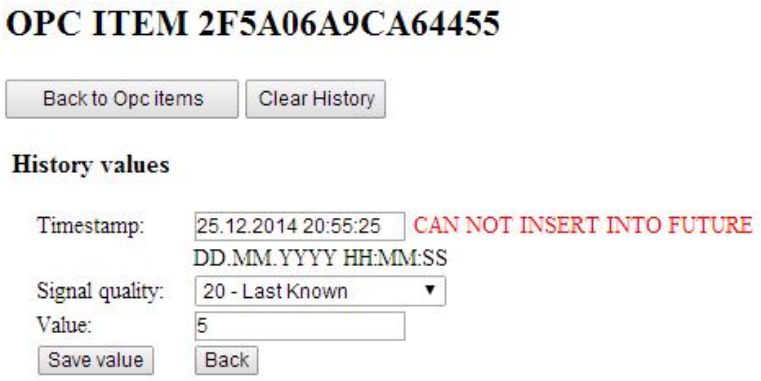
Obrázek 4: Rychlá změna hodnot na hlavní stránce

Detailní stránka

Vytvořil jsem detailní stránku pro zobrazení jednotlivých dat (OpcItems) a jejich hodnot. Na stránce jsou rozmístěny prvky pro úpravu OpcItem i jejich aktuálních hodnot, které jsou jako výchozí nastaveny tak, aby nebylo možné do nich vepisovat. Na stránce jsou také přidána tlačítka umožňující uživateli úpravu OpcItem, vrácení změn a také uložení OpcItem, při kterém je vytvořena i její výchozí aktuální hodnota, jež má hodnotu rovnu 0 a kvalitu signálu rovnu 8, které odpovídá zpráva *Not Connected*. Stejnou možnost úprav a ukládání má uživatel při úpravě aktuální hodnoty, ale navíc má možnost právě ukládanou aktuální hodnotu vložit zároveň i do historie.

Na stránce jsou také v GridView zobrazeny historické hodnoty jedné OpcItem, které jsou při velkém množství v GridView stránkovány. Historické hodnoty jsou zobrazeny podle módu historie. Když je historie povolena, ale neobsahuje žádná data pro konkrétní OpcItem, je vypsána zpráva informující, že je historie povolena, ale zároveň je prázdná. V případě, že je historie zakázaná, je o této skutečnosti uživatel informován taktéž formou zprávy a jestli je historie zakázaná, ale obsahuje nějaká data, uživatel je informován o stavu historie a ve zprávě vidí i počet historických hodnot uložených v databázi. V případě, že má daná OpcItem nějaké historické hodnoty, pak jí je může uživatel všechny smazat kliknutím na jedno tlačítko nebo může mazat historické hodnoty po jedné. Detailní stránka je zobrazena na obrázku 7.

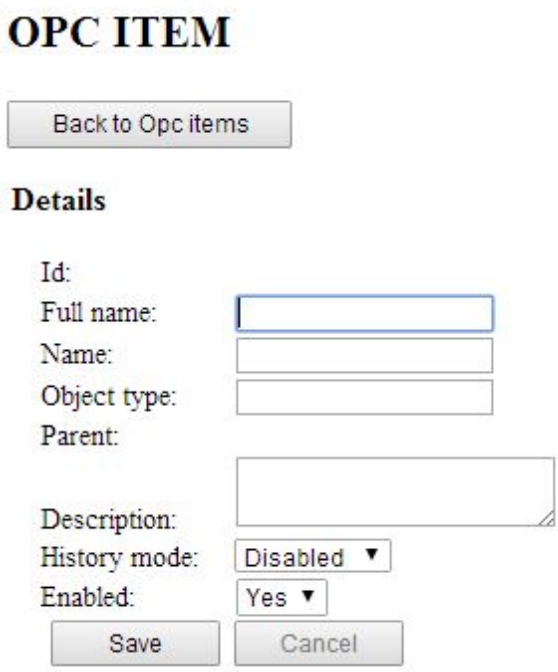
Byt'se nejspíše nebude následující funkcionalita často používat, bylo potřebné ji naimplementovat. Umožnil jsem vytvoření nové historické hodnoty. Všechny ovládací prvky na stránce jsou pro přehlednost uspořádány v panelech. Když chce uživatel vytvořit novou historickou hodnotu, všechny panely, kromě panelu s historickými hodnotami, jsou skryty, ale je skryto i GridView zobrazující historické hodnoty. Na stránce zůstanou pouze nové prvky pro vytvoření historické hodnoty. Časové razítko uživatel vepíše ručně a v případě, že by vepsal stejnou časovou hodnotu, která je již v databázi uložena, ostatní hodnoty tohoto původního záznamu se pouze aktualizují, ale nevytvoří se nový záznam v databázi. Taktéž uživatel nemá možnost vložit hodnotu do budoucnosti a je o této skutečnosti informován, což je viditelné na obrázku 5. Po vložení jsou prvky na stránce zobrazeny opět jako před vkládáním.



Obrázek 5: Vytvoření historické hodnoty

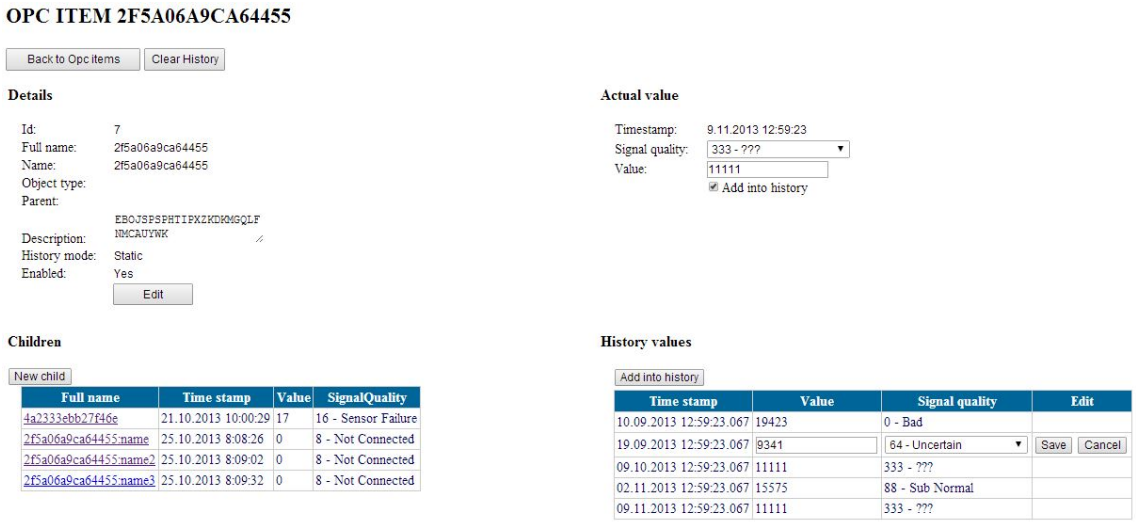
Umožnil jsem také změnu historický hodnot přímo v prvku GridView. Změna je udělána podobně jako na hlavní stránce u aktuálních hodnot, ale zde má uživatel možnost měnit hodnoty pouze po jedné, nikoliv najednou. Názorná ukázka je zobrazena na obrázku 7. Při práci bylo nepohodlné přecházet z hlavní stránky na detailní, přepisováním URL, proto jsem umožnil přecházení skrze tlačítka.

Z hlavní stránky má uživatel možnost vytvořit novou OpcItem. Funkcionalita vytváření nové Opcitem není nijak složitá. Uživatel je přesměrován na detailní stránku bez předání *Id* do URL a po přechodu jsou skryty všechny panely kromě panelu pro úpravu OpcItem, jehož prvky jsou prázdné s možností vepisovat a v danou chvíli slouží pro vytvoření OpcItem, což je viditelné na obrázku 6. Protože bych umožnění vepisování do prvků vytvářel podruhé, nadefinoval jsem enumerátor obsahující výčet možných stavů a metodu, která přijímá jako parametr hodnotu enumerátoru a podle ní pomocí konstrukce **switch** – **case** provádí změnu vlastností prvků na stránce. Volání této metody jsem ušetřil mnoho zbytečných řádků kódu a také se kód stal mnohem přehlednějším.



Obrázek 6: Vytvoření nové OpcItem

Na stránce je i druhé GridView, ve kterém jsou zobrazeni potomci právě prohlížené OpcItem. Jména potomků v GridView jsou odkazy. Když uživatel klikne na některého z potomků, pak je znovu přesměrován na detailní stránku, ve které jsou mu zobrazeny informace o právě kliknutém potomkovi i s jeho potomky. Mezi prvky pro editaci OpcItem je jméno jejího rodiče jako odkaz, skrze který je uživatel přesměrován zpět na rodičovskou OpcItem. Takto má uživatel možnost procházet stromovou hierarchií od kořene až k listům. Uživatel zde má také možnost vytvořit novou OpcItem jako potomka právě prohlížené OpcItem. Kvůli funkcionalitě stromové hierarchie bylo potřebné upravit databázi, kde jsem do jedné z tabulek přidal nové atributy a na jeden z nich jsem vytvořil index, jelikož všechny dotazy na databázi vybírající data pro stromovou hierarchii jsou spjaty s tímto atributem. Urychlil jsem takto výběr dat z databáze. Detailní stránka se zobrazenými potomky je vyobrazena na obrázku 7.



Obrázek 7: Detailní stránka se stromovou hierarchií a úpravou historické hodnoty

Implementace získávání dat přes WCF

Abych mohl zpracovávat data a předávat je aplikacím skrze službu WCF, bylo nezbytné přidanou sadu datových tříd s kontrakty nakonfigurovat, s čímž mi pomohl konzultant. Po nakonfigurování jsem začal pracovat na samotné implementaci. Ve třídě služby jsem implementoval metody, jež jsem měl předdefinovány. Začal jsem metodami pro získání jedné a více OpcItem.

Podmínky, kdy aplikace připojená k DataBrokeru z něj sbírá data, nejsou optimální. Aby bylo možné nasimulovat reálné podmínky při připojení k DataBroker Emulatoru, vytvořil jsem ve třídě konfigurace služby této aplikace dvě nové veřejné metody. Obě metody slouží ke zpoždění zaslání dat službou, ale jedna je použita pro zpoždění OpcItems a jejich aktuálních hodnot a druhá je použita pro zpoždění historických hodnot. Metody jsou volány ve službě DataBroker Emulatoru a jejich zpoždění závisí na odstupu zasílání jednotlivých OpcItems nastaveném v konfiguračním souboru připojených kontraktů krát počet samotných OpcItems.

Vytvoření slovníku s kvalitou signálu a její zobrazení

Doposud byla v GridView zobrazena kvalita signálu pouze jako číselná hodnota a rozbalovací seznam na detailní stránce byl naplněn staticky, což nebylo nejlepší řešení. Mým úkolem bylo zobrazit kvalitu signálu slovně, nikoliv její číselnou hodnotu. Vytvořil jsem statickou třídu, v jejímž konstruktoru je vytvořen slovník, který je zde prvky také naplněn. Ve třídě jsem vytvořil i metody pro výběr jednoho a více prvků ze slovníku.

Na detailní stránce jsem vymazal všechny položky rozbalovacího seznamu a seznam jsem naplnil dynamicky v metodě jeho události onInit. Řešení je zobrazeno ve výpisu 2 zdrojového kódu.

```
protected void ddlSignalQuality_Init(object sender, EventArgs e)
        {
            foreach (KeyValuePair<int, string> kvp in SignalQualityDictionary.
                GetAllSignalQualities())
            {
                int key = kvp.Key;
                string value = kvp.Value;
                ddlSignalQuality.Items.Add(new ListItem(key + " - " + value, key.ToString()));
            }
        }
```

Výpis 2: Dynamické plnění rozbalovacího seznamu

Přesun aplikace do IIS

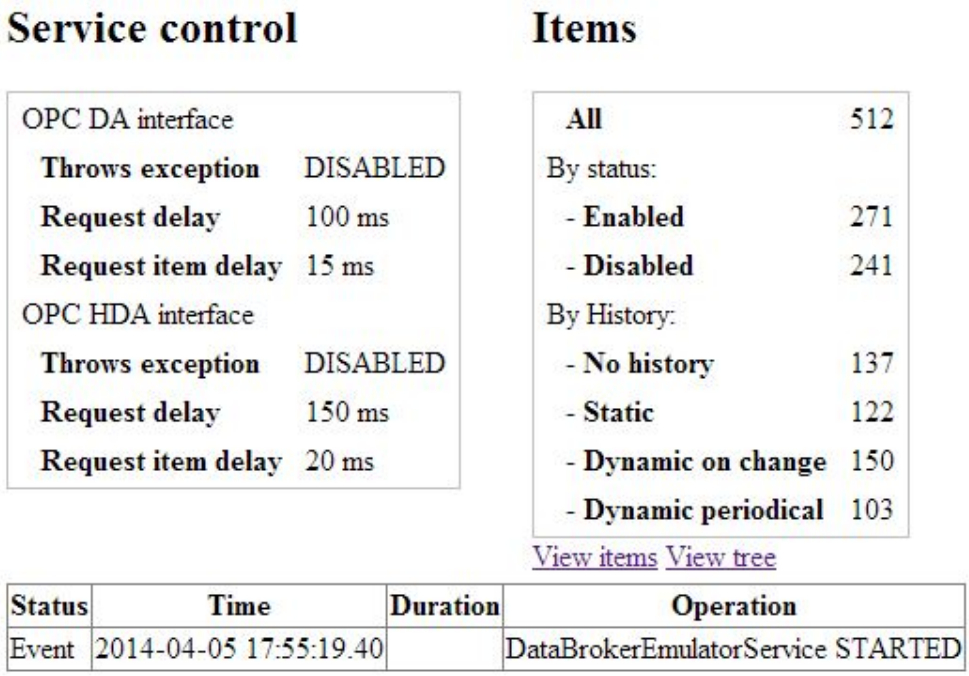
Při vývoji jsem začal mít problémy při spouštění aplikace, kdy se aplikace spustila na lokálním serveru IISExpress programu VisualStudio. Měl jsem také problémy s připojením aplikace testovací klient ke službě, protože to již lokální server nezvládal. Jelikož jsem to doposud nedělal, konzultant mi pomohl s přesunem aplikace do regulérního IIS v mém počítači. IIS mám nastaven na automatické spuštění při startu počítače, tudíž jsem při dalším vývoji nemusel spouštět celou aplikaci, ale pouze jsem nechal řešení sestavit a změny byly viditelné na stránce běžící na IIS.

Generátor dat

Doposud se mi při vyvíjení několikrát stalo, že jsem omylem upravil některý RoundhousE skript například omylem přidáním mezery. Poté již RoundhousE při spuštění záhlásil tuto skutečnost a já jsem byl nucen smazat celou databázi i s daty. Abych měl pro práci v aplikaci nějaká data, vždy jsem jich několik přidával ručně, což zabralo dost času. Proto jsem do řešení přidal nový projekt, jenž jsem nazval TestDataGenerator. Je to jednoduchá desktopová aplikace, která funguje tak, že do textového pole zadám počet prvků, které chci vygenerovat a po kliknutí jsou v cyklu náhodně vygenerována data, která jsou vložena do databáze, a nakonec je vypsána zpráva, zdali jsou data vygenerována či ne.

Vytvoření souhrnné stránky

Do projektu jsem přidal novou stránku, v níž jsem rozmístil prvky popisků. Vybral jsem počty prvků v jednotlivých módech z databáze a hodnoty vrácené dotazem na databázi jsem zobrazil na stránce, společně s informací, kdy byla aplikace spuštěna. Tato stránka je zobrazena jako první při připojení k aplikaci, proto jsem zde přidal odkaz k přechodu na hlavní stránku i na stránku stromové hierarchie. Stránka je zobrazena na obrázku 8.



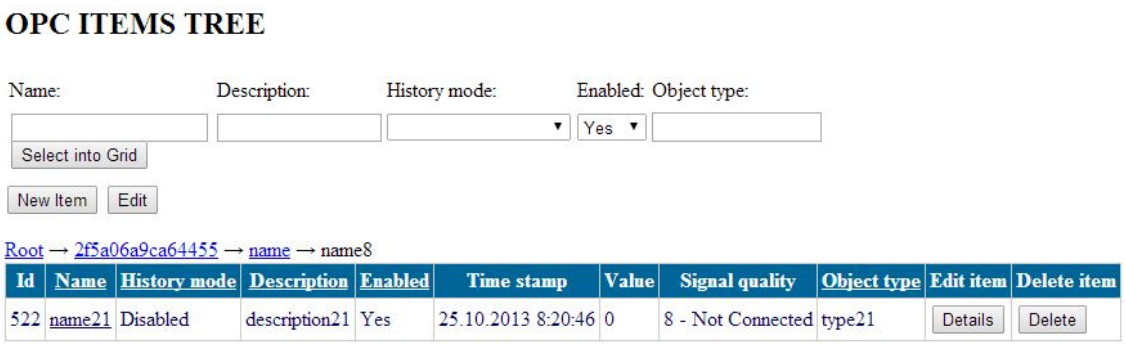
Obrázek 8: Souhrnná stránka

Stránka stromové hierarchie

Jelikož by se přidáním stromové hierarchie na hlavní stránku stala hlavní stránka nepřehledná, vytvořil jsem novou stránku určenou čistě pro procházení stromovou hierarchií s možností filtrování.

Abych nevytvářel druhé naprosto stejné GridView, vytvořil jsem si vlastní ovládací prvek GridView, který obsahuje všechnu funkcionalitu GridView z hlavní stránky a použil jsem jej na stránce nové i na hlavní stránce. Do mého GridView jsem postupně přesouval veškerou funkcionalitu původního GridView z hlavní stránky. Proces přesouvání zabral dost času. Funkcionality jsem přesouval postupně jednu po druhé, aby se nestalo, že bych se nakonec v kódu vůbec neorientoval a nic by nefungovalo jak má. Přesunul jsem HTML kód ovládacího prvku a dále jsem přesouval implementace metod zpracovávající události do nově vytvořených metod ve třídě mého GridView. Stejně tak jsem zde přesunul i implementaci metod zpracovávající události kliknutí na tlačítka, taktéž i přidané ovládací prvky DataSource a SortExtender, skrze které jsou data v GridView zobrazena a tříděna. Abych měl možnost tuto funkcionalitu používat při akcích prováděných na stránce, musel jsem ji propagovat do třídy stránky. Proto jsem vytvořil vlastní události a v metodách zpracovávajících události GridView jsou vyvolány mnou vytvořené události, jejichž metody jsou ve třídě stránky a až v těchto metodách je zpracována funkcionalita specifická pro konkrétní stránku (např. uložení indexu stránky nebo směr a atribut třídění do Session).

Po zprovoznění mého GridView na hlavní stránce jsem vytvořil novou stránku pro zobrazení stromové struktury, která vypadá skoro stejně jako hlavní stránka. Tabulka na stránce zobrazuje všechny OpcItems, ale při kliknutí na jednu z OpcItems jsou v tabulce zobrazeny pouze její potomci. Aby se uživatel orientoval, nad tabulkou je při průchodu stromovou strukturou vytvářena cesta, v níž může uživatel kliknout na kterýkoliv prvek, a budou mu v tabulce zobrazeni právě jeho potomci. Stránka s právě procházenými potomky je zobrazena na obrázku 9.



Obrázek 9: Stránka stromové hierarchie

5.3 Aplikace testovací klient

Aplikace testovacího kilenta slouží k otestování mnou vyvíjené aplikace DataBroker Emulator a měla také sloužit pro otestování aplikace DataBroker. Po spuštění aplikace si uživatel vybere, k jaké aplikaci se chce připojit a podle toho si nadefinuje adresu koncového bodu. Uživatel nemá možnost pracovat s aplikací, pokud není připojen ke službě. Po připojení si uživatel může v jedné záložce načíst OpcItems s jejími aktuálními hodnotami a v druhé záložce si může načíst historické hodnoty v časovém rozmezí, či výsledky agregačních funkcí provedených nad těmito hodnotami.

Aktuální hodnoty

Záložka pro zobrazení aktuálních hodnot funguje tak, že uživatel do textového pole vepíše názvy jednotlivých OPCItem oddělené čárkou a aplikace se připojí skrze WCF službu k aplikaci DataBroker Emulator, která mu vrátí informace o požadovaných OPCItems a jejich aktuální hodnoty. V případě, že vepsané jméno neodpovídá žádné OPCItem, záznam v tabulce testovacího klienta bude obsahovat pouze vepsané jméno a bude mít zaškrtnuté políčko Missing. Funkcionalitu jsem naimplementoval následovně. V řešení jsem vytvořil nový projekt typu Windows formulář. Konzultant mi pomohl přidat odkazy na WCF službu aplikace DataBroker Emulator, jelikož jsem doposud žádné aplikaci nenastavoval připojení ke službě. Při naplnění pole typu **string** hodnotami z textového pole se jmény jsem se poprvé setkal s použitím metod knihovny Linq, jež mi ukázal konzultant. Ukázka je zobrazena ve výpisu 3 zdrojového kódu.

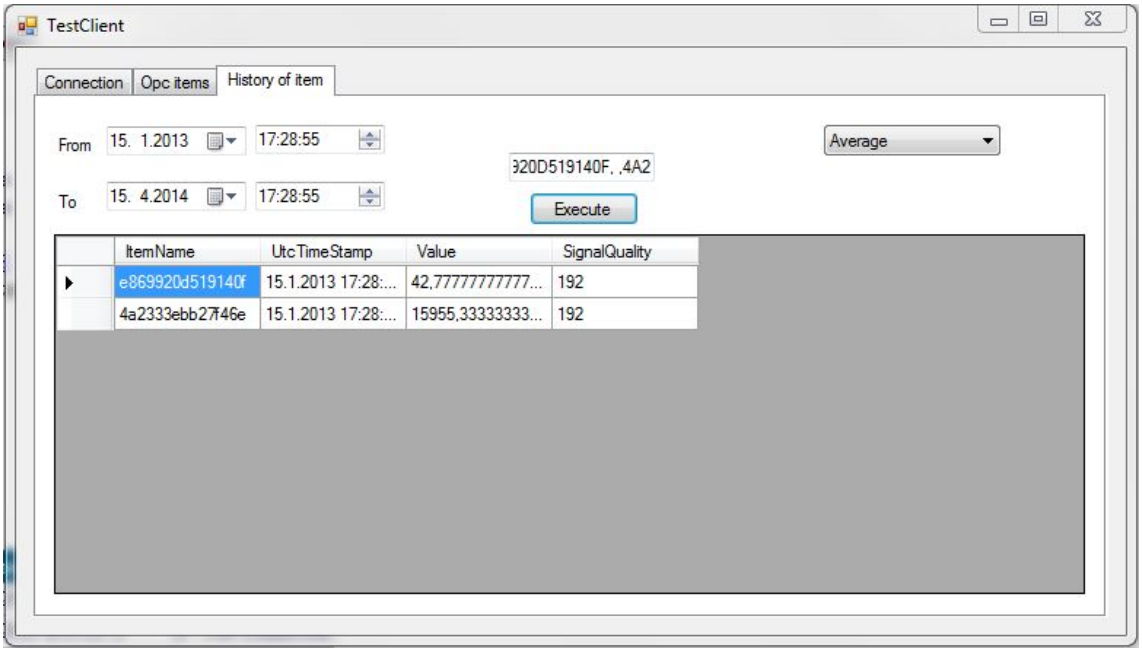
```
string[] names = txtName.Text == null ? new string[0] :
    txtName.Text.Split (',') .Where(x => !string.IsNullOrWhiteSpace(x)).Select(x => x.Trim()).
        ToArray();
```

Výpis 3: Naplnění pole jmen za použití knihovny Linq

Po získání jmen z textového pole jsou jména předána metodě služby, která po zpracování vrací hodnoty jednotlivých OpcItem. V případě, že neexistují, vrací jen jejich jména. Data jsou následně zobrazena v ovládacím prvku DataGridView. Zde jsem se setkal s další zajímavostí, a to předání dat do objektu již při jeho vytvoření (ve složených závorkách konstruktoru).

Historické hodnoty včetně agregací

V aplikaci jsem vytvořil novou záložku, jejíž layout je zobrazen na obrázku 10. Tato záložka slouží pro zobrazení historických hodnot jednotlivých OpcItems a výsledků agregačních funkcí nad těmito hodnotami. Uživatel vepíše do textového pole jména jednotlivých OpcItems oddělené čárkou, jejichž historické hodnoty chce zobrazit a vybere si časové rozpětí, ve kterém chce, aby mu byly historické hodnoty zobrazeny. Po kliknutí na tlačítko jsou z prvků DataTimePicker vybrány data a časy, viz 4. Jména OpcItems z textového pole jsou vybrána stejným způsobem jako v záložce pro výběr aktuálních hodnot, viz výpis 3 zdrojového kódu. Jména jsou dále zaslána skrze metodu služby, do aplikace Databroker Emulator, v níž jsou informace o jednotlivých OpcItems vybrány z databáze a zaslány zpět službou klientovi a ten je zobrazí v prvku DataGridView.



Obrázek 10: Zobrazení historických hodnot v aplikaci testovacího klienta

```
DateTime fromDateTime = fromDatePicker.Value.Date.Add(fromTimePicker.Value.TimeOfDay);
DateTime toDateTime = toDatePicker.Value.Date.Add(toTimePicker.Value.TimeOfDay);
```

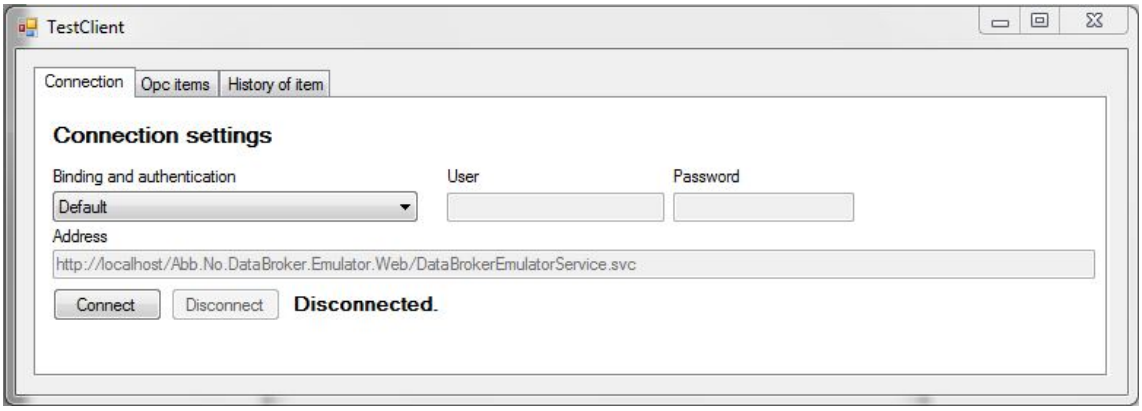
Výpis 4: Získání data a času z ovládacího prvku DateTimePicker

Taktéž má uživatel možnost zobrazit si agregované historické hodnoty. Z rozbalovacího seznamu vybere agregační funkci a po kliknutí na tlačítko je pole jmen OpcItem zasláno službě DataBroker Emulatoru, která následně vrátí historické hodnoty z databáze. Hodnoty jsou zpracovány metodou třídy *Math* odpovídající agregaci vybrané uživatelem a výsledné hodnoty jsou zobrazeny uživateli v prvku DataGridView pro každou OpcItem. V případě, že uživatel vybere rozsah hodnot, jsou hodnoty zpracovány metodami Min() a Max(), jejich výsledky jsou od sebe odečteny a výsledná hodnota je zobrazena uživateli.

Tělo metody na straně klienta, v níž se vytváří připojení ke službě, je uzavřeno v bloku příkazu **try** za nímž následuje příkaz **catch**, ve kterém jsou odchyceny případné výjimky a zobrazeny ve vyskakovacím okně. Toto odchytávání výjimek pro mě bylo velice důležité, když jsem měl v kódu nějakou chybu, protože po připojení aplikace ke službě a získávání dat z ní má služba nastaven čas, do kterého by se mělo připojení ke službě ukončit. V případě, že se spojení do této doby neukončí, služba vyvolá výjimku TimeoutException. S vyvoláváním této výjimky jsem měl problém hlavně při odstraňování chyb, kdy jsem procházel kód krok po kroku a tato výjimka mi vždy ztěžovala hledání chyby, protože jsem takto vždy překročil časový limit připojení. Proto bylo nejlepší výjimku, která hlásila mou chybu v kódu vypsat ve vyskakovacím okně dříve, než se vyvolá výjimka TimeoutException.

Možnost konfigurace připojení testovacího klienta k aplikacím DataBroker a DataBroker Emulator

Do aplikace testovací klient jsem přidal novou záložku, která se zobrazí jako první při zapnutí aplikace, jejíž layout je zobrazen na následujícím obrázku 11. Uživatel má možnost výběru způsobu připojení mezi Default, BasicHttpBinding a NetTcpBinding. V případě, že si vybere kteroukoliv možnost kromě výchozí, musí vepsat koncovou adresu služby, ke které se chce připojit. Když si uživatel vybere připojení BasicHttpBinding, vepíše koncovou adresu a klikne na tlačítko připojit, aplikace vytvoří nový objekt tohoto připojení, nastaví mu zabezpečení a připojení se vytvoří. NetTcp připojení bylo pro mě velkým oříškem. Ač by mělo být nastavení tohoto připojení podobné s připojením přes protokol http, dodnes jsem ho nevyřešil a mám tento úkol označen jako nevyřešený. Nad tímto problémem jsem strávil poměrně dlouhou dobu, časem jsem se k tomuto úkolu dodatečně vracel, ale nebylo v mých silách jej vyřešit.



Obrázek 11: Konfigurace připojení v aplikaci testovacího klienta

Z důvodu nepřehlednosti kódu jsem v aplikaci testovací klient vytvořil novou třídu s metodami, do nichž jsem přesunul mnou již napsaný zdrojový kód, který zpracovává všechnu funkcionalitu aplikace. Kód jsem postupně umazával a namísto něj jsem volal jednotlivé metody zpracovávající danou funkcionalitu z nově vytvořené třídy.

6 Complexity matrix tool

Jelikož měla firma novou zakázku, byl jsem přesunut z projektu DataBroker Emulator do začínajícího projektu nazvaného ComplexityMatrixTool. Aplikace slouží pro výpočet rizikovosti projektů v oblasti těžařství a nahradí původní soubor typu Microsoft Excel, který zákazník využíval doposud. Zákazník, pro kterého vyvíjíme projekt, je švýcarské národnosti a řídí práci spjatou s těžařstvím nerostných surovin v několika zemích světa, proto při pravidelných telefonátech a občasných setkáních jsme s ním komunikovali v anglickém jazyce. K aplikaci budou uživatelé přistupovat ve dvou rolích (manažer a administrátor), kdy uživatel v roli manažera má možnost aplikaci využívat pouze pro výpočet rizikovosti, načež administrátor může celou aplikaci nastavovat vytvářením otázek, jejich kategorií, odpovědí na otázky, jejich skupin a možnost nastavovat pravidla nadřazená výpočtům pro určené otázky. Již dříve zmíněný soubor typu Microsoft Excel jsme dostali jako zadání a sloužil nám jako předloha k aplikaci.

Na projektu jsem pracoval se studentem VŠB-TUO, který již měl bakalářskou praxi ukončenou. Jelikož se jako vedoucí projektu rozhodl pro vyvíjení projektu za pomoci technologie ASP.NET MVC a agilní metodiky SCRUM, viz reference [3], prvním úkolem bylo, nastudovat si tuto metodiku dopodrobna, taktéž jsem studoval technologii ASP.NET MVC, protože jsem s ní neměl žádné zkušenosti. Novým možnostem v této technologii se učím dodnes. Jako další úkol jsme vytvořili nový projekt a připojili jej k novému TFS. Poté jsme studovali funkčnost původního souboru typu Microsoft Excel. Když jsme nastudovali funkčnost souboru, přešli jsme k dalšímu úkolu, jehož náplní bylo navrhnout databázi a vytvořit UML diagram popisující tento návrh. Při návrhu databáze jsme se od kolegů ve firmě dozvěděli, že existuje nástroj PetaPoco, který nám velice usnadní práci a vygeneruje za nás podle databázových tabulek ORM. PetaPoco jsme do projektu doinstalovali. Do projektu jsme také přidali nástroj RoundhousE a ve skriptech přidaných do něj jsme právě navrženou databázi, sloužící pro uložení formulářových otázek, odpovědí a dalších informací vytvořili.

V tuto chvíli jsme měli základ aplikace připraven a mohli jsme se přesunout k vytváření funkcionality. Mým úkolem bylo vytvořit nové modely odpovídající databázovým tabulkám s definovanými validacemi. Poté, co jsem vypracoval modely, a byly schváleny mým spolupracovníkem, společně jsme pokračovali vytvářením CRUD operací způsobem, že každý z nás vytvářel základní CRUD operace nad odlišnými databázovými tabulkami. Až po všech připravených metodách provádějících CRUD operace jsme vytvořili Controller, v němž jsme postupně implementovali metody typu GET, abychom byli schopni vygenerovat View pro každou funkcionalitu, u níž jsme View potřebovali. Po vytvoření View jsme v Controlleru také nadefinovali metody typu POST, díky nimž máme možnost zpracovávat data na straně serveru a volat metody z modelů, pomocí nichž jsou data převedena do databázových objektů a uložena do databáze, popřípadě jiná data z databáze vybrána a skrze metodu Controlleru zaslána do View a zobrazena, či metody obsahující jinou funkcionalitu, ale týkající se pouze objektu daného modelu. Při vytváření View jsme využívali značkovací syntaxi Razor, která nám umožnila použití zdrojového kódu programovacího jazyka C# při vytváření náhledu stránky. Pomocí

této technologie jsme schopni dynamicky generovat prvky stránky či ovládací prvky na stránce. V této době se k naší práci připojil nový student z Finska využívající programu Erasmus, tudíž jsem byl přirozeně nucen s ním také řešit problémy týkající se projektu a jediným jazykem, jakým jsme se spolu mohli dorozumět, byla angličtina, tudíž jsem takto zlepšoval i svou jazykovou schopnost. Mým dalším úkolem bylo upravit validace v modelech, které obsahují vlastnost jméno, jelikož jsem musel pomocí regulárního výrazu omezit možnost vložení nepřijatelných znaků či prázdných znaků do pole jména a pole váhy bylo nutno omezit pouze pro vložení desetinných čísel v rozmezí 0 až 1.

Další velkou novinkou pro mne bylo použití JavaScript, konkrétně knihovny JQuery, pomocí níž jsme následně vytvářeli výpočty pro výslednou tabulku s ohodnocením rizik a pro paprskový graf zobrazující míru rizika. Všechny výpočty a zobrazení jsou přepočítávány a prováděny dynamicky při výběru každé z odpovědí, aniž by byla data ze stránky předána serveru ke zpracování. Mým posledním úkolem zahrnující odborná praxe bylo vytvořit uživatelský manuál, který je pochopitelně psán v anglickém jazyce, jelikož bude aplikace používána ve více zemích světa.

7 Získané a scházející znalosti a dovednosti v průběhu odborné praxe

S první neznámou při absolvování odborné praxe jsem se setkal hned první pracovní den, jelikož jsem se poprvé dozvěděl o agilní metodice vývoje SCRUM. O této metodice jsem se ve škole učil až v předmětu Vývoj informačních systémů, který jsem však vykonal až po odpracování odborné praxe. Také pro mne bylo novinkou vývoj aplikace v týmu.

Další velkou neznámou pro mne byla celkově služba WCF, kterou jsem musel před použitím nastudovat a jejíž základy mi vysvětlil konzultant. Se službou obecně a připojením aplikací ke službě jsem se rovněž setkal až v předmětu Vývoj informačních systémů.

V dalším projektu, na jehož vývoji jsem se podílel, pro mne byla úplně nová věc - technologie ASP.NET MVC. Z předmětu Databázové a informační systémy jsem měl o této technologii základní teoretické povědomí, ovšem prakticky jsem se s touto technologií setkal až při vykonávání praxe.

Stejnou novinkou pro mne byl skriptovací jazyk JavaScript společně s jeho knihovnou JQuery, o nichž jsem věděl, že existují, ale do vykonání odborné praxe jsem se nesetkal s jejich implementací.

Když se však ohlédnu na druhou stranu, neměl jsem při vykonání odborné praxe žádný větší problém s pracemi s databází. Z předmětu databázové a informační systémy jsem si také odnesl základní znalosti vývoje informačního systému, které jsem již na praxi pouze rozšiřoval.

Podobně tomu je i celkově u jazyka C#, jehož základy jsem měl dobře vžity již ze školy a na praxi jsem si znalosti pouze rozšiřoval.

8 Závěr

V této práci jsem se pokusil vystihnout veškerou práci, kterou jsem prováděl po dobu mé odborné praxe. Odborná praxe pro mne byla velice přínosná. Jsem moc rád, že jsem ji mohl vykonat právě ve firmě ABB s.r.o., v níž jsem mohl vyvíjet aplikace po boku dlouholetých a zkušenostmi nabitých programátorů. Kdykoliv, když jsem si nevěděl rady s nějakou implementací, mohl jsem se obrátit na svého konzultanta nebo na některého z jeho kolegů, kteří mi velice přátelsky poskytli radu či návrh, jak daný problém řešit, což pro mne často byla neocenitelná zkušenost.

V předchozí kapitole jsem popsal technologie a techniky, které jsem se v průběhu odborné praxe musel doučit, ovšem dobré základy mám vybudovány ze školy. Na VŠB-TUO jsem nastoupil po absolvování všeobecného gymnázia a byť jsem se lehce zajímal o programování, v té době jsem neměl ani tušení, jak to ve světě programování vlastně funguje a co všechno se člověk musí naučit, aby psal čistý kód. Tímto bych chtěl říci, že jsem se v průběhu studia naučil mnoho věcí, ale myslím si, že v tomto odvětví se člověk může novým technologiím a technikám učit celý život.

Jiří Littner

9 Reference

[1] ABB GROUP: *Abb group - Automation and Power Technologies. [online]. [cit. 2014-04-16].* Dostupné z: http://www.abb.com

[2] MICROSOFT DEVELOPER NETWORK: *Web and .NET Development. [online]. [cit. 2014-04-19].* Dostupné z: http://msdn.microsoft.com/library

[3] SCHWABER, Ken a Jeff SUTHERLAND. *SCRUM Guide. [online]. [cit. 2014-04-21].* Dostupné z: https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf

[4] GOOGLE PROJECT: *RoundhousE. [online]. [cit. 2014-04-21].* Dostupné z: https://code.google.com/p/roundhouse/#"Professional_Database_Versioning_and_Change_Management&quo

[5] TOPTENSOFTWARE: *PetaPoco. [online]. [cit. 2014-04-21].* Dostupné z: http://www.toptensoftware.com/petapoco/

[6] STACK OVERFLOW: *Stack Overflow. [online]. [cit. 2014-04-21].* Dostupné z: http://www.stackoverflow.com/